LTL Control in Uncertain Environments with Probabilistic Satisfaction Guarantees - Technical Report -

Xu Chu Ding Stephen L. Smith Calin Belta Daniela Rus

***Abstract*— We present a method to generate a robot control strategy that maximizes the probability to accomplish a task. The task is given as a Linear Temporal Logic (LTL) formula over a set of properties that can be satisfied at the regions of a partitioned environment. We assume that the probabilities with which the properties are satisfied at the regions are known, and the robot can determine the truth value of a proposition only at the current region. Motivated by several results on partitioned-based abstractions, we assume that the motion is performed on a graph. To account for noisy sensors and actuators, we assume that a control action enables several transitions with known probabilities. We show that this problem can be reduced to the problem of generating a control policy for a Markov Decision Process (MDP) such that the probability of satisfying an LTL formula over its states is maximized. We provide a complete solution for the latter problem that builds on existing results from probabilistic model checking. We include an illustrative case study.**

I. Introduction

Recently there has been an increased interest in using temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) as motion specification languages for robotics [1]–[6]. Temporal logics are appealing because they provide formal, high level languages in which to describe complex missions, *e.g.,* "Reach A, then B, and then C, in this order, infinitely often. Never go to A. Don't go to B unless C or D were visited." In addition, off-the-shelf model checking algorithms [7], [8] and temporal logic game strategies [9] can be used to verify the correctness of robot trajectories and to synthesize robot control strategies.

Motivated by several results on finite abstractions of control systems, in this paper we assume that the motion of the robot in the environment is modeled as a finite labeled transition system. This can be obtained by simply partitioning the environment and labeling the edges of the corresponding quotient graph according to the motion capabilities of the robot among the regions. Alternatively, the partition can be made in the state space of the robot dynamics, and the transition system is then a finite abstraction of a continuous or hybrid control system [10], [11].

The problem of controlling a finite transition system from a temporal logic specification has received a lot of attention during recent years. All the existing works assume that the current state can be precisely determined. If the result of a control action is deterministic (*i.e.,* at each state, an available control enables exactly one transition), control strategies from specifications given as LTL formulas can be found through a simple adaptation of off-the-shelf model checking algorithms [3]. If the control is nondeterministic (an available control at a state enables one of several transitions, and their probabilities are not known), the control problem from an LTL specification can be mapped to the solution of a Rabin game [12], or simpler Büchi and GR(1) games if the specification is restricted to fragments of LTL [1]. If the control is probabilistic (an available control at a state enables one of several transitions, and their probabilities are known), the transition system is a Markov Decision Process (MDP). The control problem then reduces to generating a policy (adversary) for an MDP such that the produced language satisfies a formula of a probabilistic temporal logic [13], [14]. We have recently developed a framework for deriving an MDP control strategy from a formula in a fragment of probabilistic CTL (pCTL) [15]. For probabilistic LTL, in [16], a control strategy is synthesized for an MDP where some states are under control of the environment, so that an LTL specification is guaranteed to be satisfied under all possible environment behaviors. The temporal logic control problems for systems with probabilistic or nondeterministic state-observation models, which include the class of Partially Observable Markov Decision Processes [17], [18], are currently open.

In this paper, we consider motion specifications given as arbitrary LTL formulas over a set of properties that can be satisfied with given probabilities at the vertices of a graph environment. We assume that the truth values of the properties can be observed only when a vertex is reached in the environment, and the observations of these properties are independent with each other. We assume a probabilistic robot control model and that the robot can determine its current vertex precisely. Under these assumptions, we develop an algorithm to generate a control strategy that maximizes the probability of satisfying the specification. Our approach is based on mapping this problem to the problem of generating a control policy for a MDP such that the probability of satisfying an LTL formula is maximized. We provide a solution to this problem by drawing inspiration from probabilistic model checking. We illustrate the method by applying it to a numerical example of a robot navigating in an indoor environment.

The contribution of this work is twofold. First, we adapt

This work was supported in part by ONR-MURI N00014-09-1051, ARO W911NF-09-1-0088, AFOSR YIP FA9550-09-1-020, and NSF CNS-0834260.

X. C. Ding and C. Belta are with Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA (email: {xcding; cbelta}@bu.edu). S. L. Smith is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (email: stephen.smith@uwaterloo.ca). D. Rus is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA (email: rus@csail.mit.edu).

existing approaches in probabilistic model checking (*e.g.,* [19], [20]), and provide a complete solution to the general problem of controlling MDPs from full LTL specifications using deterministic Rabin automata. This is a significant departure from our previous work on MDP control from pCTL formulas [15], since it allows for strictly richer specifications. The increase in expressivity is particularly important in many robotic applications where the robot is expected to perform some tasks, such as surveillance, repeatedly. However, it comes at the price of increased computational complexity. Second, we allow for non-determinism not only in the robot motion, but also in the robot's observation of properties in the environment. This allows us to model a large class of robotic problems in which the satisfaction of properties of interest can be predicted only probabilistically. For example, we can model a task where a robot is operating in an indoor environment, and is required to pick-up and deliver items among some rooms. The robot determines its current location using RFID tags on the floors and walls. Non-determinism occurs in observations because items may or may not be available when a robot visits a room. Non-determinism also occurs in the motion due to imprecise localization or control actuation.

The remainder of the paper is organized as follows: In Section II we introduce the necessary definitions and preliminary results. In Section III we formulate the problem and describe the technical approach. In Section IV we reformulate this problem onto a MDP and show that two problems are equivalent. We synthesis our controls strategy in Section V, and an example of the provided algorithm is shown in Section VI. We conclude in Section VII.

II. Preliminaries

In this section we provide background material on linear temporal logic and Markov decision processes.

A. Linear Temporal Logic

We employ Linear Temporal Logic (LTL) to describe high level motion specifications. A detailed description of the syntax and semantics of LTL is beyond the scope of this paper and can be found in, for example, [7]. Roughly, an LTL formula is built up from a set of atomic propositions Π, which are properties that can be either true or false, standard Boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), and temporal operators $\bigcirc$ (next), $\mathcal{U}$ (until), $\Diamond$ (eventually), $\Box$ (always) and $\Rightarrow$ (implication). The semantics of LTL formulas are given over words, which is defined as an infinite sequence $o = o_0 o_1 \ldots$, where $o_i \in 2^{\Pi}$ for all i.

We say $o \vDash \phi$ if the word o satisfies the LTL formula ϕ. The semantics of LTL is defined recursively. If $\phi = \pi$ is an LTL formula, where $\pi \in \Pi$, then ϕ is true at position i of the word if $\pi \in o_i$. A word satisfies an LTL formula ϕ if ϕ is true at the first position of the word; $\Box\phi$ means that ϕ is true at all positions of the word; $\Diamond\phi$ means that ϕ eventually becomes true in the word; $\phi_1 \,\mathcal{U}\phi_2$ means ϕ_2 eventually becomes true and ϕ_1 is true until this happens; $\bigcirc\phi$ means that ϕ becomes true at next position of the word. More expressivity can be achieved by combining the above temporal and Boolean operators (several examples are given later in the paper). An LTL formula can be represented by a deterministic Rabin automaton, which is defined as follows.

Definition 2.1 (Deterministic Rabin Automaton): A deterministic Rabin automaton (DRA) is a tuple $\mathcal{R} = (Q, \Sigma, \delta, q_0, F)$, where (i) Q is a finite set of states; (ii) Σ is a set of inputs (alphabet); (iii) $\delta : Q \times \Sigma \to Q$ is the transition function; (iv) $q_0 \in Q$ is the initial state; and (v) $F = \{(L_1, K_1), \ldots, (L_k, K_k)\}$ is a set of pairs where $L_i, K_i \subseteq Q$ for all $i \in \{1, \ldots, k\}$.

A run of a Rabin automaton $\mathcal{R}$, denoted by $r_{\mathcal{R}} = q_0 q_1 \ldots$, is an infinite sequence of states in $\mathcal{R}$ such that for each $i \geq 0$, $q_{i+1} \in \delta(q_i, \alpha)$ for some $\alpha \in \Sigma$. A run $r_{\mathcal{R}}$ is *accepting* if there exists a pair $(L, K) \in F$ such that 1) there exists $n \geq 0$, such that for all $m \geq n$, we have $q_m \notin L$, and 2) there exist infinitely many indices k where $q_k \in K$. This acceptance conditions means that $r_{\mathcal{R}}$ is accepting if for a pair $(L, K) \in F$, $r_{\mathcal{R}}$ intersects with L finitely many times and K infinitely many times.

For any LTL formula ϕ over Π, one can construct a DRA with input alphabet $\Sigma = 2^{\Pi}$ accepting all and only words over Π that satisfy ϕ (see [21]). We refer readers to [22] and references therein for algorithms and to freely available implementations, such as [23], to translate a LTL formula over Π to a corresponding DRA.

B. Markov Decision Process and probability measure

We now introduce a labeled Markov decision process, and the probability measure we will use in the upcoming sections.

Definition 2.2 (Labeled Markov Decision Process): A labeled Markov decision process (MDP) is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{U}, \mathcal{A}, \mathcal{P}, \iota, \Pi, h)$, where (i) $\mathcal{S}$ is a finite set of states; (ii) $\mathcal{U}$ is a finite set of actions; (iii) $\mathcal{A} : \mathcal{S} \to 2^{\mathcal{U}}$ represents the set of actions enabled at state $s \in \mathcal{S}$; (iv) $\mathcal{P} : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to [0, 1]$ is the transition probability function such that for all states $s \in \mathcal{S}$, $\sum_{s' \in \mathcal{S}} \mathcal{P}(s, u, s') = 1$ if $u \in \mathcal{A}(s) \subseteq \mathcal{U}$ and $\mathcal{P}(s, u, s') = 0$ if $u \notin \mathcal{A}(s)$; (v) $\iota : \mathcal{S} \to [0, 1]$ is the initial state distribution satisfying $\sum_{s \in \mathcal{S}} \iota(s) = 1$; (vi) Π is a set of atomic propositions; and (vii) $h : \mathcal{S} \to 2^{\Pi}$ is a labeling function.

The quantity $\mathcal{P}(s, u, s')$ represents the probability of reaching the state s' from s taking the control $u \in \mathcal{A}(s)$.

We will now define a probability measure over paths in the MDP. To do this, we define an action function as a function $\mu : \mathcal{S} \to \mathcal{U}$ such that $\mu(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}$. An infinite sequence of action functions $M = \{\mu_0, \mu_1, \ldots\}$ is called a policy. One can use a policy to resolve all nondeterministic choices in an MDP by applying the action $\mu_k(s_k)$ at each time-step k. Given an initial state s_0 such that $\iota(s_0) > 0$, an infinite sequence $r^M_{\mathcal{M}} = s_0 s_1 \ldots$ on $\mathcal{M}$ generated under a policy $M = \{\mu_0, \mu_1, \ldots\}$ is called a path on $\mathcal{M}$ if $\mathcal{P}(s_i, \mu_i(s_i), s_{i+1}) > 0$ for all i. The subsequence $s_0 s_1 \ldots s_n$ is called a finite path. If $\mu_i = \mu$ for all i, then we call this policy a stationary policy.

We define $\mathrm{Paths}^M_{\mathcal{M}}$ and $\mathrm{FPaths}^M_{\mathcal{M}}$ as the set of all infinite and finite paths of $\mathcal{M}$ under a policy M starting from any state s_0 where $\iota(s_0) > 0$. We can then define a probability measure over the set $\mathrm{Paths}^M_{\mathcal{M}}$ of paths. For a

path $r^M_{\mathcal{M}} = s_0 s_1 \dots s_n s_{n+1} \dots \in \mathrm{Paths}^M_{\mathcal{M}}$, the *prefix* of length n of $r^M_{\mathcal{M}}$ is the finite subsequence $s_0 s_1 \dots s_n$. Let $\mathrm{Paths}^M_{\mathcal{M}}(s_0 s_1 \dots s_n)$ denote the set of all paths in $\mathrm{Paths}^M_{\mathcal{M}}$ with the prefix $s_0 s_1 \dots s_n$. (Note that $s_0 s_1 \dots s_n$ is a finite path in $\mathrm{FPaths}^M_{\mathcal{M}}$.)

Then, the probability measure $\Pr^M$ on the smallest σ-algebra over $\mathrm{Paths}^M_{\mathcal{M}}$ containing $\mathrm{Paths}^M_{\mathcal{M}}(s_0 s_1 \dots s_n)$ for all $s_0 s_1 \dots s_n \in \mathrm{FPaths}^M_{\mathcal{M}}$ is the unique measure satisfying

$$\begin{aligned} & \Pr^M\{\mathrm{Paths}^M_{\mathcal{M}}(s_0 s_1 \dots s_n)\} \\ = \; & \iota(s_0) \prod_{0 \le i < n} \mathcal{P}(s_i, \mu_i(s_i), s_{i+1}). \end{aligned} \tag{1}$$

Finally, we can define the probability that a policy M in an MDP $\mathcal{M}$ satisfies an LTL formula ϕ. A path $r^M_{\mathcal{M}} = s_0 s_1 \dots$ deterministically generates a word $o = o_0 o_1 \dots$ where $o_i = h(s_i)$ for all i. With a slight abuse of notation, we denote $h(r^M_{\mathcal{M}})$ as the word generated by $r^M_{\mathcal{M}}$. Given an LTL formula ϕ, one can show that the set $\{r^M_{\mathcal{M}} \in \mathrm{Paths}^M_{\mathcal{M}} : h(r^M_{\mathcal{M}}) \vDash \phi\}$ is measurable. We define

$$\Pr^M_{\mathcal{M}}(\phi) := \Pr^M\{r^M_{\mathcal{M}} \in \mathrm{Paths}^M_{\mathcal{M}} : h(r^M_{\mathcal{M}}) \vDash \phi\} \tag{2}$$

as the probability of satisfying ϕ for $\mathcal{M}$ under policy M. For more details about probability measures on MDPs under a policy and measurability of LTL formulas, we refer the reader to a text in probabilistic model checking, such as [19].

III. Model, Problem Formulation, and Approach

In this section we formalize the environment model, the robot motion model, and the robot observation model. We then formally state our problem and provide a summary of our technical approach.

A. Environment, task, and robot model

1) Environment model: In this paper, we consider a robot moving in a partitioned environment, which can be represented by a graph and a set of properties:

$$\mathcal{E} = (V, \delta_{\mathcal{E}}, \Pi), \tag{3}$$

where V is the set of vertices, $\delta_{\mathcal{E}} \subseteq V \times V$ is the relation modeling the set of edges, and Π is the set of properties (or atomic propositions). Such a finite representation of the environment can be obtained by using popular partition schemes, such as triangulations or rectangular grids. The set V can be considered as a set of labels for the regions in the partitioned environment, and $\delta_{\mathcal{E}}$ is the corresponding adjacency relation. In this paper we assume that there is no blocking vertex in V (*i.e.,* all vertices have at least one outgoing edge).

2) Task specification: The atomic propositions Π represent properties in the environment that can be true of false. We require the motion of the robot in the environment to satisfy a rich specification given as an LTL formula ϕ over Π (see Sec. II). A variety of robotic tasks can be easily translated to LTL formulas. For example,

- Parking: "Find parking lot and then park" $((\Diamond \text{parking lot}) \wedge (\text{parking lot} \Rightarrow \bigcirc \text{park}))$
- Data Collection: "Always gather data at gathering locations and then upload the data, repeat infinitely many times" $(\Box\Diamond(\text{gather} \Rightarrow \Diamond \text{upload}))$
- Ensure Safety: "Achieve task ψ while always avoiding states satisfying P_1 or P_2" $(\Box\neg(P_1 \vee P_2) \wedge \psi)$.

3) Robot motion model: The motion capability of the robot in the environment is represented by a set of motion primitives U, and a function $A : V \to 2^U$ that returns the set of motion primitives available (or enabled) at a vertex $v \in V$. For example, U can be $\{\text{Turn Left}, \text{Turn Right}, \text{Go Straight}\}$ in an urban environment with roads and intersections. To model non-determinism due to possible actuation or measurement errors, we define the transition probability function $P_m : V \times U \times V \to [0, 1]$ such that $\sum_{v' \in V} P_m(v, u, v') = 1$ for all $v \in V$ and $u \in A(v)$, and $P_m(v, u, v') = 0$ if $(v, v') \notin \delta_{\mathcal{E}}$ or if $u \notin A(v)$. Thus, $P_m(v, u, v')$ is the probability that after applying the motion primitive u at vertex v, the robot moves from v to an adjacent region v' without passing through other regions. The set U corresponds to a set of feedback controllers for the robot. Such feedback controllers can be constructed from facet reachability (see [24], [25]), and the transition probabilities can be obtained from experiments (see [15]). Note that this model of motion uses an underlying assumption that transition probabilities of the robot controllers do not depend on the previous history of the robot.

4) Robot observation model: In our earlier work [3], we assumed that the motion of the robot in the partitioned environment is deterministic, and we proposed an automatic framework to produce a provably correct control strategy so that the trajectory of the robot satisfies an LTL formula. In [15], we relaxed this restriction and allowed non-determinism in the motion of the robot, and a control strategy for the robot was obtained to maximize the probability of satisfying a task specified by a fragment of CTL. In both of these results, it was assumed that some propositions in Π are associated with each region in the environment (*i.e.,* for each $v \in V$), and they are fixed in time.

However, this assumption is restrictive and often not true in practice. For example, the robot might move to a road and find it congested; while finding parking spots, some parking spots may already be taken; or while attempting to upload data at an upload station, the upload station might be occupied. We wish to design control strategies that react to information which is observed in real-time, *e.g.,* if a road is blocked, then pick another route.

Motivated by these scenarios, in this paper we consider the problem setting where observations of the properties of the environment are probabilistic. To this end, we define a probability function $P_o : V \times \Pi \to [0, 1]$. Thus, $P_o(v, \pi)$ is the probability that the atomic proposition $\pi \in \Pi$ is observed at a vertex $v \in V$ when v is visited. We assume that all observations of atomic propositions for a vertex $v \in V$ are independent and identically distributed. This is a reasonable model in situations where the time-scale of robot travel is larger than the time scale on which the proposition changes. For future work, we are pursuing more general observation

models. Let $\Pi_v := \{\pi \in \Pi : P_o(v,\pi) > 0\}$ be the atomic propositions that can be observed at a vertex v. Then $Z_v = \{Z \in 2^{\Pi_v} : \prod_{\pi \in Z} P_o(v,\pi) \times \prod_{\pi \notin Z} (1 - P_o(v,\pi)) > 0\}$ is the set of all possible observations at v.

B. Problem Formulation

Let the initial state of the robot be given as v_0. The trajectory of the robot in the environment is an infinite sequence $r = v_0 v_1, \ldots$, where $P_m(v_i, u, v_{i+1}) > 0$ for some u for all i. Given $r = v_0 v_1, \ldots$, we call v_i the state of the robot at the discrete time-step i. We denote the observed atomic propositions at time-step i as $o_i \in Z_{v_i}$ and $O(r) = o_0 o_1 \ldots$ as the word observed by r. An example of a trajectory r and its observed word in an environment with given $\mathcal{E}$, U, A, P_m and P_o are shown in Fig. 1.

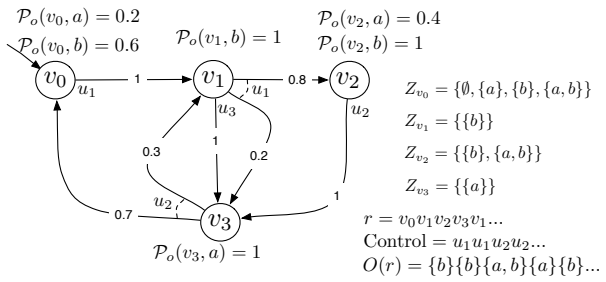

Fig. 1. An example trajectory r and its observed word $O(r)$. We also show Z_v for all $v \in V$. A single arrow pointed towards a state v_0 indicates the initial state. The atomic proposition set is $\Pi = \{a, b\}$. The set of motion primitives is $U = \{u_1, u_2, u_3\}$. The probability function P_o assigns probabilities for all atomic propositions at each state. We show the probability of an atomic proposition only if it is positive (*i.e.,* $\pi \in \Pi_v$). The number on top of an arrow pointing from a vertex v to v' is the probability $P_m(v, u, v')$ associated with a control $u \in U$.

Our desired "reactive" control strategy is in the form of an infinite sequence $C = \{\nu_0, \nu_1, \ldots\}$ where $\nu_i : V \times 2^{\Pi} \to U$ and $\nu_i(v, Z)$ is defined only if $Z \in Z_v$. Furthermore, we enforce that $\nu_i(v, Z) \in A(v)$ for all v and all i. The reactive control strategy returns the control to be applied at each time-step, given the current state v and observed set of propositions Z at v. Given an initial condition v_0 and a control strategy C, we can produce a trajectory $r = v_0 v_1 \ldots$ where the control applied at time i is $\nu_i(v_i, o_i)$. We call r and $O(r) = o = o_0 o_1 \ldots$ the trajectory and the word generated under C, respectively. Note that given v_0 and a control strategy C, the resultant trajectory and its corresponding word are not unique due to non-determinism in both motion and observation of the robot.

Now we formulate the following problem:

Problem 3.1: Given the environment represented by $\mathcal{E} = (V, \delta_{\mathcal{E}}, \Pi)$; the robot motion model U, A and P_m; the observation model P_o; and an LTL formula ϕ over Π, find the control strategy C that maximizes the probability that the word generated under C satisfies ϕ.

C. Summary of technical approach

Our approach to solve Prob. 3.1 proceeds by construction of a labeled MDP $\mathcal{M}$ (see Def. 2.2), which captures all possible words that can be observed by the robot. Furthermore, each control strategy C corresponds uniquely to a policy M on $\mathcal{M}$. Thus, each trajectory with an observed word under a control strategy C corresponds uniquely to a path on $\mathcal{M}$ under M. We then reformulate Prob. 3.1 as the problem of finding the policy on $\mathcal{M}$ that maximizes the probability of satisfying ϕ. These two problems are equivalent due to the assumption that all observations are independent. We synthesize the optimal control strategy by solving maximal reachability probability problems inspired by results in probabilistic model checking. Our framework is more general than in [15] due to a richer specification language and non-determinism in observation of the environment. The trade off is that computational complexity in this approach is in general much larger due to increased size of the automaton representing the specification.

IV. MDP Construction and Problem Reformulation

As part of our approach to solve Problem 3.1, we construct a labeled MDP $\mathcal{M} = (\mathcal{S}, \mathcal{U}, \mathcal{A}, \mathcal{P}, \iota, \Pi, h)$ from the environment model $\mathcal{E}$, the robot motion model U, A, P_m, and the observation model P_o as follows:

- $\mathcal{S} = \{(v, Z) \,|\, v \in V, Z \in Z_v\}$
- $\mathcal{U} = U$
- $\mathcal{A}((v, Z)) = A(v)$
- $\mathcal{P}((v, Z), u, (v', Z')) =$

$$P_m(v, u, v') \times \left(\prod_{\pi \in Z'} P_o(v', \pi) \times \prod_{\pi \notin Z'} (1 - P_o(v', \pi)) \right)$$

- ι is defined as $\iota(s) = \prod_{\pi \in Z} \mathcal{P}(v_0, \pi) \times \prod_{\pi \notin Z} (1 - \mathcal{P}(v_0, \pi))$ if $s = (v_0, Z)$ for any $Z \in Z_{v_0}$, and $\iota(s) = 0$ otherwise.
- $h((v, Z)) = Z$ for all $(v, Z) \in S$.

An example of a constructed MDP is shown in Fig. 2. One can easily verify that $\mathcal{M}$ is a valid MDP such that for all $s \in \mathcal{S}$, $\sum_{s' \in \mathcal{S}} \mathcal{P}(s, u, s') = 1$ if $u \in \mathcal{A}(s)$, $\mathcal{P}(s, u, s') = 0$ if $u \notin \mathcal{A}(s)$, and $\sum_{s \in \mathcal{S}} \iota(s) = 1$. We discuss the growth of the state space from $\mathcal{E}$ to $\mathcal{M}$ in Section V-C.

We now formulate a problem on the MDP $\mathcal{M}$. We will then show that this new problem is equivalent to Prob. 3.1.

Problem 4.1: For a given labeled MDP $\mathcal{M}$ and an LTL formula ϕ, find a policy such that $\Pr^M_{\mathcal{M}}(\phi)$ (see Eq. (2)) is maximized.

The following proposition formalizes the equivalence between the two problems, and the one-to-one correspondence between a control strategy on $\mathcal{E}$ and a policy on $\mathcal{M}$.

Proposition 4.2 (Equivalence of problems): A control strategy $C = \{\nu_0, \nu_1, \ldots\}$ is a solution to Problem 3.1 if and only if the policy $M = \{\mu_0, \mu_1, \ldots\}$, where

$$\mu_i\big((v_i, Z_i)\big) = \nu_i(v_i, Z_i) \quad \text{for each } i,$$

is a solution to Problem 4.1.

Proof: We can establish an one-to-one correspondence between a control strategy C in the environment $\mathcal{E}$ and a policy M on $\mathcal{M}$. Given $C = \{\nu_0, \nu_1, \ldots\}$, we can obtain the corresponding $M = \{\mu_0, \mu_1, \ldots\}$ by setting $\mu_i((v_i, Z_i)) = \nu_i(v_i, Z_i)$. Conversly, given $M = \{\mu_0, \mu_1, \ldots\}$, we can

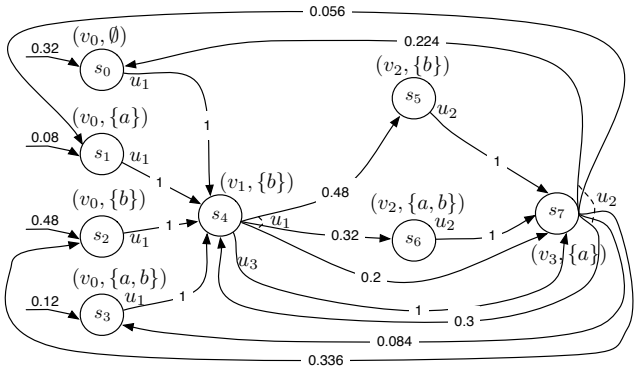

Fig. 2. The constructed MDP $\mathcal{M}$ using $\mathcal{E}$, U, A, P_m and P_o from the example in Fig. 1. For each state $s \in \mathcal{S}$, the labels on top of the state show the components of s (*i.e.,* $s = (v, Z)$). The number on the arrow from the state (v, Z) to the state (v', Z') denotes the transition probability $\mathcal{P}((v, Z), u, (v', Z'))$ for the action $u \in \mathcal{U}$. The numbers atop arrows pointing into states (v_0, Z_{v_0}) denote the initial distribution. The set of atomic propositions assigned to each state in $\mathcal{M}$ is the second component of the state.

generate a corresponding control strategy $C = \{\nu_0, \nu_1, \ldots\}$ such that $\nu_i(v_i, Z_i) = \mu_i((v_i, Z_i))$.

We need only to verify that we can use the same probability measure on paths in $\mathcal{E}$ and on trajectories in $\mathcal{M}$. Due to the assumption that each observation at v is independent, the observation process is Markovian as it only depends on which vertex the observation is made. Note that the probability of observing $Z \in Z_v$ at a state $v \in V$ is $\prod_{\pi \in Z} \mathcal{P}(v, \pi) \times \prod_{\pi \notin Z} (1 - \mathcal{P}(v, \pi))$. Hence, the probability of moving from a vertex v to v', under control u, and observing $Z' \in Z_{v'}$ is exactly $\mathcal{P}((v, Z), u(v', Z'))$. Therefore, the probability of observing the finite word $O(r^f)$ for a finite trajectory $r^f = v_0 \ldots v_n$ under C is the same (by construction of $\mathcal{M}$) as traversing through a finite path $fr^M_{\mathcal{M}} \in \mathrm{FPaths}^M_{\mathcal{M}}$ such that $h(fr^M_{\mathcal{M}}) = O(r^f)$ under the policy M corresponding to C. Since this property holds for any arbitrary finite trajectory r^f, a trajectory r with a word $O(r)$ under C can be uniquely mapped to a path in $\mathrm{Paths}^M_{\mathcal{M}}$ and we can use the probability measure and σ-algebra (see Sec. II-B) on $\mathcal{M}$ under a policy M for the corresponding control strategy C. Thus, if M is a solution for Prob. 3.1, then the control strategy C corresponding to M is a solution for Prob. 4.1, and vice versa. ∎

∎

Due to the above proposition, we will proceed by constructing a policy M on the MDP $\mathcal{M}$ as a solution to Prob. 4.1. We can then uniquely map M to a control strategy C in the robot environment $\mathcal{E}$ for a solution to Prob. 3.1.

V. Synthesis of Control Strategy

In this section we provides a solution for Prob. 3.1 by synthesizing an optimal policy for Prob. 4.1. Our approach is adapted from automata-theoretic approaches in the area of probabilistic verification and model checking (see [20] and references therein for an overview). Probabilistic LTL model checking finds the maximum probability that a path of a given MDP satisfies a LTL specification. We modify this method to obtain an optimal policy that achieves the maximum probability. This approach is related to the work of [26], in which rewards are assigned to specifications and nondeterministic Büchi automata (NBA) are used. We do not use NBAs since a desired product MDP cannot be directly constructed from an NBA, but only from an DRA.

A. The Product MDP

We start by converting the LTL formula ϕ to a DRA defined in Def. 2.1. We denote the resulting DRA as $\mathcal{R}_\phi = (Q, 2^\Pi, \delta, q_0, F)$ with $F = \{(L_1, K_1), \ldots, (L_k, K_k)\}$ where $L_i, K_i \subseteq Q$ for all $i = 1, \ldots, k$. The DRA obtained from the LTL formula $\phi = \Box\Diamond a \wedge \Box\Diamond b$ is shown in Fig. 3.

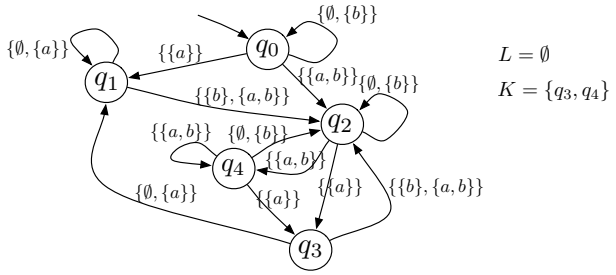

Fig. 3. The DRA $\mathcal{R}_\phi$ corresponding to the LTL formula $\phi = \Box\Diamond a \wedge \Box\Diamond b$. In this example, there is one set of accepting states $F = \{(L, K)\}$ where $L = \emptyset$ and $K = \{q_3, q_4\}$. Thus, accepting runs of this DRA must visit q_3 or q_4 (or both) infinitely often.

We now obtain an MDP as the product of a labeled MDP $\mathcal{M}$ and a DRA $\mathcal{R}_\phi$. This product MDP allows one to find runs on $\mathcal{M}$ that generate words satisfying the acceptance condition of $\mathcal{R}_\phi$.

Definition 5.1 (Product MDP): The product MDP $\mathcal{M} \times \mathcal{R}_\phi$ between a labeled MDP $\mathcal{M} = (\mathcal{S}, \mathcal{U}, \mathcal{A}, \mathcal{P}, \iota, \Pi, h)$ and a DRA $\mathcal{R}_\phi = (Q, 2^\Pi, \delta, q_0, F)$ is a MDP $\mathcal{M}_\mathcal{P} = (\mathcal{S}_\mathcal{P}, \mathcal{U}, \mathcal{A}_\mathcal{P}, \mathcal{P}_\mathcal{P}, \iota_\mathcal{P})$, where:

- $\mathcal{S}_\mathcal{P} = \mathcal{S} \times Q$ (the Cartesian product of sets $\mathcal{S}$ and Q)
- $\mathcal{A}_\mathcal{P}((s, q)) = \mathcal{A}(s)$
- $\mathcal{P}_\mathcal{P}((s, q), u, (s', q')) =$

$$\begin{cases} \mathcal{P}(s, u, s') & \text{if } q' = \delta(q, h(s')) \\ 0 & \text{otherwise} \end{cases}$$

- $\iota_\mathcal{P}((s, q)) = \iota(s)$ if $q = \delta(q_0, h(s))$ and $\iota_\mathcal{P} = 0$ otherwise.

We generate the accepting state pairs $F_\mathcal{P}$ for the product MDP $\mathcal{M}_\mathcal{P}$ as follows: For a pair $(L_i, K_i) \in F$, a state (s, q) of $\mathcal{M}_\mathcal{P}$ is in $L^\mathcal{P}_i$ if $q \in L_i$, and $(s, q) \in K^\mathcal{P}_i$ if $q \in K_i$.

As an example, we show in Fig. 4 some of the states and transitions for the product MDP $\mathcal{M}_\mathcal{P} = \mathcal{M} \times \mathcal{R}_\phi$ where $\mathcal{M}$ is shown in Fig. 2 and $\mathcal{R}_\phi$ is shown in Fig. 3.

Note that the set of actions for $\mathcal{M}_\mathcal{P}$ is the same as the one for $\mathcal{M}$. A policy $M_\mathcal{P} = \{\mu^\mathcal{P}_0, \mu^\mathcal{P}_1, \ldots\}$ on $\mathcal{M}_\mathcal{P}$ directly induces a policy $M = \{\mu_0, \mu_1, \ldots\}$ on $\mathcal{M}$ by keeping track of the state on the product MDP ($\mu^\mathcal{P}_i$ is an action function that returns an action corresponding to a state in $\mathcal{M}_\mathcal{P}$). Note

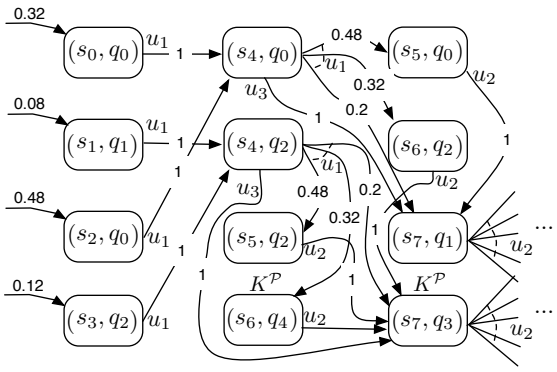

Fig. 4. The product MDP $\mathcal{M}_{\mathcal{P}} = \mathcal{M} \times \mathcal{R}_{\phi}$ where $\mathcal{M}$ is shown in Fig. 2 and $\mathcal{R}_{\phi}$ is shown in Fig. 3. Due to space restrictions, only the initial states (states for which the initial distribution $\iota_{\mathcal{P}}((s,q))$ is positive) and part of the state space are shown. $F_{\mathcal{P}} = \{(L^{\mathcal{P}}, K^{\mathcal{P}})\}$, where $L^{\mathcal{P}} = \emptyset$ and states in $K^{\mathcal{P}}$ are marked.

that given the state of $\mathcal{M}$ at time-step i and the state of $\mathcal{M}_{\mathcal{P}}$ at time-step $i-1$, the state of $\mathcal{M}_{\mathcal{P}}$ at time-step i can be exactly determined. We can induce a policy M for $\mathcal{M}$ from a policy $M_{\mathcal{P}}$ for $\mathcal{M}_{\mathcal{P}}$ as follows:

Definition 5.2 (Inducing a policy for $\mathcal{M}$ from $\mathcal{M}_{\mathcal{P}}$): If the state of $\mathcal{M}_{\mathcal{P}}$ at time-step i is (s_i, q_i), then the policy $M = \{\mu_0, \mu_1, \ldots\}$ induced from $M_{\mathcal{P}} = \{\mu_0^{\mathcal{P}}, \mu_1^{\mathcal{P}} \ldots\}$ can be obtained by setting $\mu_i(s_i) = \mu_i^{\mathcal{P}}((s_i, q_i))$ for all i.

We denote $r_{\mathcal{M}_{\mathcal{P}}}^{M_{\mathcal{P}}}$ as a path on $\mathcal{M}_{\mathcal{P}}$ under a policy $M_{\mathcal{P}}$. We say a path $r_{\mathcal{M}_{\mathcal{P}}}^{M_{\mathcal{P}}}$ is accepting if and only if it satisfies the Rabin acceptance condition with $F_{\mathcal{P}}$ as the accepting states pairs, *i.e.,* there exists a pair $(L^{\mathcal{P}}, K^{\mathcal{P}}) \in F_{\mathcal{P}}$ so that $r_{\mathcal{M}_{\mathcal{P}}}^{M_{\mathcal{P}}}$ intersects with $L^{\mathcal{P}}$ finitely many times and $K^{\mathcal{P}}$ infinitely many times.

The product MDP is constructed so that given a path $r_{\mathcal{M}_{\mathcal{P}}}^{M_{\mathcal{P}}} = (s_0, q_0)(s_1, q_1) \ldots$, the path $s_0 s_1 \ldots$ on $\mathcal{M}$ generates a word that satisfies ϕ if and only if the infinite sequence $q_0 q_1 \ldots$ is an accepting run on $\mathcal{R}_{\phi}$, in which case $r_{\mathcal{M}_{\mathcal{P}}}^{M_{\mathcal{P}}}$ is accepting. Therefore, each accepting path of $\mathcal{M}_{\mathcal{P}}$ uniquely corresponds to a paths of $\mathcal{M}$ whose word satisfies ϕ.

B. Generating the Optimal Control Strategy

Once we obtain the product MDP $\mathcal{M}_{\mathcal{P}} = \mathcal{M} \times \mathcal{R}_{\phi}$ and the accepting states pairs $F_{\mathcal{P}} = \{(L_1^{\mathcal{P}}, K_1^{\mathcal{P}}), \ldots, (L_k^{\mathcal{P}}, K_k^{\mathcal{P}})\}$, the method to obtain a solution to Prob. 3.1 proceeds as follows: For each pair $(L_i^{\mathcal{P}}, K_i^{\mathcal{P}}) \in F_{\mathcal{P}}$, we obtain a set of accepting maximum end components. An accepting maximum end component for $\mathcal{M}_{\mathcal{P}}$ consists of a set of states $\overline{\mathcal{S}_{\mathcal{P}}} \subseteq \mathcal{S}_{\mathcal{P}}$ and a function $\overline{\mathcal{A}_{\mathcal{P}}}$ such that $\emptyset \neq \overline{\mathcal{A}_{\mathcal{P}}}((s,q)) \subseteq \mathcal{A}_{\mathcal{P}}((s,q))$ for all $(s,q) \in \overline{\mathcal{S}_{\mathcal{P}}}$. It has the property that, by taking actions enabled by $\overline{\mathcal{A}_{\mathcal{P}}}$, all states in $\overline{\mathcal{S}_{\mathcal{P}}}$ can reach every other state in $\overline{\mathcal{S}_{\mathcal{P}}}$ and can not reach any state outside of $\overline{\mathcal{S}_{\mathcal{P}}}$. Furthermore, it contains no state in $L_i^{\mathcal{P}}$ and at least one state in $K_i^{\mathcal{P}}$. In addition, it is called maximum because it is not properly contained in another accepting maximum end component. Note that for a given pair $(L_i^{\mathcal{P}}, K_i^{\mathcal{P}})$, its accepting maximal end components are pairwise disjoint.

A procedure to obtain all accepting maximum end components of an MDP is outlined in [19]. From probabilistic model checking (see [19], [20]), the maximum probability of satisfying the LTL formula ϕ for $\mathcal{M}$ is the same as the maximum probability of reaching any accepting maximum end component of $\mathcal{M}_{\mathcal{P}}$. Once an accepting maximum end component $(\overline{\mathcal{S}_{\mathcal{P}}}, \overline{\mathcal{A}_{\mathcal{P}}})$ is reached, then all states in $\overline{\mathcal{S}_{\mathcal{P}}}$ are reached infinitely often (and ϕ satisfied) with probability 1, under a policy that all actions in $\overline{\mathcal{A}_{\mathcal{P}}}$ are used infinitely often.

The maximum probability of reaching a set of states $B_{\mathcal{P}} \subseteq \mathcal{S}_{\mathcal{P}}$ can be obtained by the solution of a linear program. First we find the set of states that can not reach $B_{\mathcal{P}}$ under any policy and denote it as $C_{\mathcal{P}}$ (a simple graph analysis is sufficient to find this set). We then let x_p denote the maximum probability of reaching the set $B_{\mathcal{P}}$ from a state $p \in \mathcal{S}_{\mathcal{P}}$. We have $x_p = 1$ if $p \in B_{\mathcal{P}}$, and $x_p = 0$ if $p \in C_{\mathcal{P}}$. The values of x_p for the remaining states can then be determined by solving a linear optimization problem:

$$\begin{aligned} &\min \sum_{p \in S_{\mathcal{P}}} x_p, \text{ subject to: } 0 \leq x_p \leq 1, \text{ and} \\ &x_p \geq \sum_{t \in S_{\mathcal{P}}} \mathcal{P}_{\mathcal{P}}(p,u,t) x_t \text{ for all } p \in S_{\mathcal{P}} \setminus (B_{\mathcal{P}} \cup C_{\mathcal{P}}) \\ &\text{and for all } u \in \mathcal{A}_{\mathcal{P}}(p). \end{aligned} \tag{4}$$

Once x_p is obtained for all $p \in S_{\mathcal{P}}$, one can identify an action $u^{\star}$ (not necessarily unique) for each state $p \in S_{\mathcal{P}} \setminus (B_{\mathcal{P}} \cup C_{\mathcal{P}})$ such that:

$$x_p = \sum_{t \in S_{\mathcal{P}}} \mathcal{P}_{\mathcal{P}}(p, u^{\star}, t) x_t. \tag{5}$$

We define a function $\mu_{\mathcal{P}}^{\star} : \mathcal{S}_{\mathcal{P}} \to \mathcal{U}$ that returns an action $u^{\star}$ satisfying (5) if $p \in S_{\mathcal{P}} \setminus (B_{\mathcal{P}} \cup C_{\mathcal{P}})$ (the actions for states in $B_{\mathcal{P}}$ or $C_{\mathcal{P}}$ are irrelevant and can be chosen arbitrarily). Then the optimal policy maximizing the probability of reaching $B_{\mathcal{P}}$ is the stationary policy $\{\mu_{\mathcal{P}}^{\star}, \mu_{\mathcal{P}}^{\star}, \ldots\}$.

Our desired policy $M_{\mathcal{P}}^{\star}$ that maximizes the probability of satisfying ϕ on the product MDP is the policy maximizing the probability of reaching the union of all accepting maximum end components of all accepting state pairs in $F_{\mathcal{P}}$, if the state is not in an accepting maximum end component. Otherwise, the optimal policy is to use all actions allowed in the associated accepting maximal end component infinitely often in a round-robin fashion. The solution to Prob. 4.1 is then the policy $M^{\star}$ on $\mathcal{M}$ induced by $M_{\mathcal{P}}^{\star}$. The desired reactive control strategy $C^{\star}$ as a solution to Prob. 3.1 can finally be obtained as the control strategy corresponding to $M^{\star}$ (see Prop. 4.2). Our overall approach is summarized in Alg. 1.

C. Complexity

The complexity of our proposed algorithm is dictated by the size of the generated MDPs. We use $|\cdot|$ to denote cardinality of a set. The number of states in $\mathcal{M}$ is $|\mathcal{S}| = \sum_{v \in V} |Z_v|$. Hence, in the worst case where all propositions $\pi \in \Pi$ can be observed with positive but less than 1 probability at all vertices $v \in V$, $|\mathcal{S}| = 2^{|\Pi|}$. In practice, the number of propositions that can be non-deterministically observed at a vertex is small. For example, in an urban setting, most regions of the environment including intersections and roads

Algorithm 1 Generating the optimal control strategy $C^\star$ given $\mathcal{E}$, U, A, P_m, P_o and ϕ

1: Generate the MDP $\mathcal{M}$ from the environment model $\mathcal{E}$, the motion primitives U, the actions A, the motion model P_m and the observation model P_o.
2: Translate the LTL formula ϕ to a deterministic Rabin automaton $\mathcal{R}_\phi$.
3: Generate the product MDP $\mathcal{M}_\mathcal{P} = \mathcal{M} \times \mathcal{R}_\phi$ and accepting states pairs $F_\mathcal{P} = \{(L_1^\mathcal{P}, K_1^\mathcal{P}), \ldots, (L_k^\mathcal{P}, K_k^\mathcal{P})\}$.
4: Find all accepting maximum end components for all pairs $(L_i^\mathcal{P}, K_i^\mathcal{P}) \in F_\mathcal{P}$, and find their union $B_\mathcal{P}$.
5: Find the stationary policy $\{\mu_\mathcal{P}^\star, \mu_\mathcal{P}^\star, \ldots\}$ maximizing the probability of reaching $B_\mathcal{P}$ by solving (4) and (5).
6: Generate the policy $M_\mathcal{P}^\star = \{\mu_0^\mathcal{P}, \mu_1^\mathcal{P}, \ldots\}$ as follows: $\mu_i^\mathcal{P}(p) = \mu_\mathcal{P}^\star(p)$ if $p \in \mathcal{S}_\mathcal{P} \setminus B_\mathcal{P}$. Otherwise, p is in at least one accepting maximum end component. Assuming it is $(\overline{\mathcal{S}_\mathcal{P}}, \overline{\mathcal{A}_\mathcal{P}})$ and $\overline{\mathcal{A}_\mathcal{P}}(p) = \{u_1, u_2, \ldots, u_m\}$, then $\mu_i^\mathcal{P}(p) = u_j$ where $j = i \bmod m$.
7: Generate the policy $M^\star = \{\mu_0, \mu_1, \ldots\}$ induced by $M_\mathcal{P}^\star$.
8: Generate the control strategy $C^\star = \{\nu_0, \nu_1, \ldots\}$ corresponding to $M^\star$ by setting $\nu_i(v, Z) = \mu_i((v, Z))$ for all i.

have fixed atomic propositions. The number of intersections that can be blocked is small comparing to the size of the environment.

The size of the DRA $|Q|$ is in worst case, doubly exponential with respect to $|\Pi|$. However, empirical studies such as [22] have shown that in practice, the sizes of the DRAs for many LTL formulas are exponential or lower with respect to $|\Pi|$. In robot control applications, since properties in the environment are typically assigned scarcely (meaning that each region of the environment is usually assigned a small number of properties comparing to $|\Pi|$), the size of DRA can be reduced much further by removing transitions in the DRA with inputs that can never appear in the environment, and then minimizing the DRA by removing states that can not be reached from the initial state.

The size of the product MDP $\mathcal{M}_\mathcal{P}$ is $|\mathcal{M}| \times |Q|$. The complexity for the algorithm to generate accepting maximal end component is at most quadratic in the size of $\mathcal{M}_\mathcal{P}$ (see [19]), and the complexity for finding the optimal policy from a linear program is polynomial in the size of $\mathcal{M}_\mathcal{P}$. Thus, overall, our algorithm is polynomial in the size of $\mathcal{M}_\mathcal{P}$.

VI. Example

The computational framework developed in this paper is implemented in MATLAB, and here we provide an example as a case study. Consider a robot navigating in an indoor environment as shown in Fig. 5. Each region of the environment is represented by a vertex v_i, and the arrows represent allowable transitions between regions. In this case study, we choose the motion primitives arbitrarily (see the caption of Fig. 5). In practice, they can either correspond to low level control actions such as "turn left", "turn right" and "go straight", or high level commands such as "go from region 1 to region 2", which can then be achieved by a sequence of low level control actions.

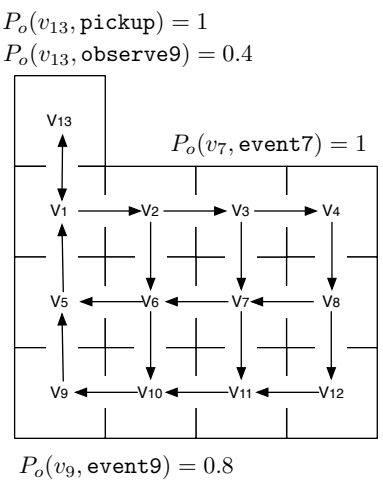

Fig. 5. Environment for a numerical example of the proposed approach. We assume that the set of motion primitive is $U = \{\alpha, \beta, \gamma\}$. The number of actions available at each vertex depends on the number of arrows from that vertex to adjacent vertices. We define the enabling function A so that the motion primitive α is enabled at all vertices, β is enabled at vertices v_1, v_6 and v_7, and γ is enabled at vertices v_2, v_3, v_6, v_7 and v_8.

The goal of the robot is to perform a persistent surveillance mission on regions v_7 and v_9, described as follows: The robot can pickup (or receive) a surveillance task at region v_{13}. With probability 0.4 the robot receives the task denoted `observe9`. Otherwise, the task is `observe7`. The task `observe7` (or `observe9`) is completed by traveling to region v_7 (or v_9), and observing some specified event. In region v_7, the robot observes the event (`event7`) with probability 1. In region v_9, each time the robot enters the region, there is a probability of 0.8 that it observes the event (`event9`). Thus, the robot may have to visit v_9 multiple times before observing `event9`. Once the robot observes the required event, it must return to v_{13} and pickup a new task.

This surveillance mission can be represented by four atomic propositions $\{\texttt{pickup}, \texttt{observe9}, \texttt{event7}, \texttt{event9}\}$. (the task `observe7` can be written as $\neg$`observe9`). The propositions `pickup` and `observe7` are assigned to v_{13}, with $P_o(v_{13}, \texttt{pickup}) = 1$ and $P_o(v_{13}, \texttt{observe9}) = 0.4$. The proposition `event7` is assigned to v_7 with $P_o(v_7, \texttt{event7}) = 1$ and `event9` is assigned to v_9 with $P_o(v_9, \texttt{event9}) = 0.8$.

The surveillance mission can be written as the following LTL formula:

$$\phi = \square\diamond\texttt{pickup} \wedge$$
$$\square\left(\texttt{pickup} \wedge \neg\texttt{observe9} \Rightarrow \bigcirc(\neg\texttt{pickup}\ \mathcal{U}\texttt{event7})\right)$$
$$\wedge\ \square\left(\texttt{pickup} \wedge \texttt{observe9} \Rightarrow \bigcirc(\neg\texttt{pickup}\ \mathcal{U}\texttt{event9})\right).$$

The first line of ϕ, $\square\diamond$`pickup`, enforces that the robot must repeatedly pick up tasks. The second line pertains to

task observe7 and third line pertains to task observe9. These two lines ensure that a new task cannot be picked up until the current task is completed (*i.e.,* the desired event is observed). Note that if event9 is observed after observing event7, then the formula ϕ is not violated (and similarly if event7 is observed after observing event9).

The MDP $\mathcal{M}$ generated from the environment is shown in Fig. 6. For this example, we have arbitrarily chosen values for the probability transition function P_m. In practice, probabilities of transition under actuation and measurement errors can be obtained from experiments or accurate simulations (see [15]). The number of states in the MDP $\mathcal{M}$ is $|S| = 15$.

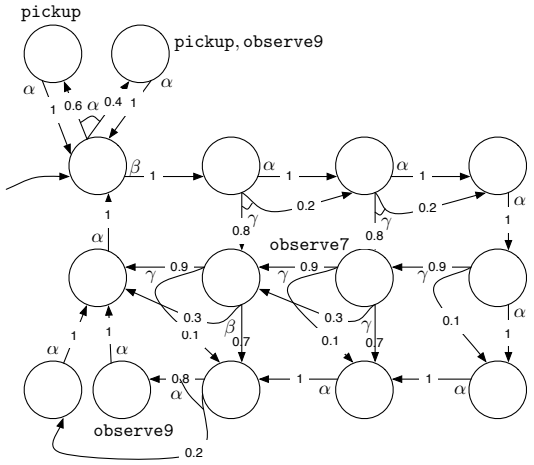

Fig. 6. MDP $\mathcal{M}$ generated from the environment with given U, A, P_o and P_m. The initial state s_0 is marked by an incoming arrow ($\iota(s_0) = 1$).

We generated the deterministic Rabin automaton $\mathcal{R}_\phi$ using the ltl2dstar tool (see [23]). The number of states $|Q|$ is 52. Thus, the product MDP $\mathcal{M}_\mathcal{P}$ has 780 states. For the DRA generated, there is only one set in F, *i.e.,* $F = \{(L, K)\}$, with 1 state in L and 18 states in K. Thus, the number of states in $L^\mathcal{P}$ is 15 and the number of states in $K^\mathcal{P}$ is 270. There is one accepting maximum end component in $\mathcal{M}_\mathcal{P}$, and it contains 17 states.

Using the implementation of Alg. 1 we computed the maximum probability of satisfying the specification from the initial state and the optimal control strategy. The Algorithm ran in approximately 7 seconds on a MacBook Pro computer with a 2.5 GHz dual core processor. For this example the maximum probability is 1, implying that the corresponding optimal control strategy almost surely satisfies ϕ. To illustrate the control strategy, a sample execution is shown in Fig. 7.

VII. Conclusions and Final Remarks

We presented a method to generate a robot control strategy that maximizes the probability to accomplish a task. The robot motion in the environment was modeled as a graph and the task was given as a Linear Temporal Logic (LTL) formula over a set of properties that can be satisfied at the vertices with some probability. We allowed for noisy sensors and actuators by assuming that a control action enables several transitions with known probabilities. We reduced this problem to one of generating a control policy for a Markov Decision Process such that the probability of satisfying an LTL formula over its states is maximized. We then provided a complete solution to this problem adapting existing probabilistic model checking tools.

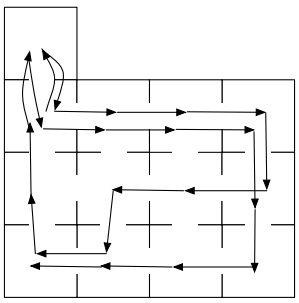

Fig. 7. A sample path of the robot with the optimal control strategy. The word observed by the sample path is pickup, event7, event9, {pickup, observe9}, event9, ...

We are currently pursuing several future directions. We are looking at proposition observation models that are not independently distributed. These models arise when the current truth value of the proposition gives information about the future truth value. We are also looking at methods for optimizing the robot control strategy for a suitable cost function when costs are assigned to actions of an MDP. The second direction will build on our recent results on optimal motion planning with LTL constraints [27].

References

[1] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, "Where's Waldo? Sensor-based temporal logic motion planning," in *IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, 2007, pp. 3116–3121.

[2] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *IEEE Conf. on Decision and Control*, Shanghai, China, 2009, pp. 2222 – 2229.

[3] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[4] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multiagent motion tasks based on LTL specifications," in *IEEE Conf. on Decision and Control*, Paradise Island, Bahamas, 2004, pp. 153–158.

[5] M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi, "Multi-robot motion planning: A timed automata approach," in *IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA, Apr. 2004, pp. 4417–4422.

[6] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *IEEE Conf. on Decision and Control*, Shanghai, China, 2009, pp. 5997–6004.

[7] E. M. Clarke, D. Peled, and O. Grumberg, *Model checking*. MIT Press, 1999.

[8] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science: Formal Models and Semantics*, J. van Leeuwen, Ed. Elsevier, 1990, vol. B, pp. 995–1072.

[9] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive(1) designs," in *International Conference on Verification, Model Checking, and Abstract Interpretation*, Charleston, SC, 2006, pp. 364–380.

[10] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, pp. 971–984, 2000.

[11] G. J. Pappas, "Bisimilar linear systems," *Automatica*, vol. 39, no. 12, pp. 2035–2047, 2003.

[12] W. Thomas, "Infinite games and verification," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, E. Brinksma and K. Larsen, Eds. Springer, 2002, vol. 2404, pp. 58–65.

[13] A. Dianco and L. D. Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Foundations of Software Technology and Theoretical Computer Science*, ser. Lecture Notes in Computer Science. Springer, 1995, vol. 1026, pp. 499–513.

[14] M. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic symbolic model checking with PRISM: A hybrid approach," *International Journal on Software Tools for Technology Transfer*, vol. 6, no. 2, pp. 128–142, 2004.

[15] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta, "Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees," in *IEEE Int. Conf. on Robotics and Automation*, Anchorage, AK, 2010, pp. 3227 – 3232.

[16] C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski, "Controller synthesis for probabilistic systems," in *Proceedings of IFIP TCS*, 2004.

[17] J. Pineau and S. Thrun, "High-level robot behavior control using POMDPs," in *AAAI Workshop notes*, Menlo Park, CA, 2002.

[18] N. L. Zhang and W. Zhang, "Speeding up the convergence of value iteration in partially observable Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 14, pp. 29–51, 2001.

[19] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[20] M. Vardi, "Probabilistic linear-time model checking: An overview of the automata-theoretic approach," *Formal Methods for Real-Time and Probabilistic Systems*, pp. 265–276, 1999.

[21] E. Gradel, W. Thomas, and T. Wilke, *Automata, logics, and infinite games: A guide to current research*, ser. Lecture Notes in Computer Science. Springer, 2002, vol. 2500.

[22] J. Klein and C. Baier, "Experiments with deterministic ω-automata for formulas of linear temporal logic," *Theoretical Computer Science*, vol. 363, no. 2, pp. 182–195, 2006.

[23] J. Klein, "ltl2dstar - LTL to deterministic Streett and Rabin automata," http://www.ltl2dstar.de/, 2007.

[24] L. Habets and J. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, no. 1, pp. 21–35, 2004.

[25] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 864–874, 2005.

[26] C. Courcoubetis and M. Yannakakis, "Markov decision processes and regular events," *IEEE Transactions on Automatic Control*, vol. 43, no. 10, pp. 1399–1418, 1998.

[27] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning under temporal constraints," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Taipei, Taiwan, Oct. 2010, to appear.